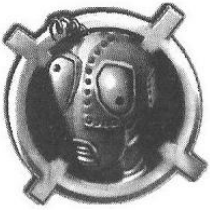
ROBO RALLY



The RoboRally Operating Manual

CREDITS

Designer: Richard Garfield
Developer and "Keeper of the Flame": Mike Davis
Art Director: Daniel Gelon
Graphic design and layout: Maria Cabardo, Daniel Gelon, Anson Maddocks, and Tom Wänerstrand
Editor: Paul Hughes
Additional editing: Kathryn Haines and Jenny Scott
Project coordination: Glenn Elliott and Paul Randles
Publishing services and print coordination: Keith Kentop, Darlene Miller, and Tom Wänerstrand
Cover art: Phil Foglio
Logo design: Daniel Gelon and Christopher Rush
Equipment renderings: Rob McLees

Robots and robot personalities designed by Phil Foglio

Playtesters: Chris Barrett, Russ and Ginger Berg, Glenn and Kathy Briscoe, Denis Brown, Bill Brum, Craig Castonguay, Dmitrc Catledge-Pine-Burns, Patty Davis, George Emery, Tom Eppright, Becky Forgy-Richardson, James Gary, Richard George, John Hina, Judy Huddell, Alex Lamb, Gary Majors, Chris McDonough, Bruce Newton, Brian and Joyce Ostberg, Barbara Roden, David Ruckstuhl, Joe Williams, Colby players, ConQuest of Santa Maria, JimCon '87, Naperville Bell Labs players, Naperville Nalco players, NCSU players, Penn bridge club players, Pensacola players, WPI players, and Richard, Lee, Susan, Justin, and Elizabeth Garfield

Thanks to the rest of the RoboRally Team (Adam Conus, Steve Schmeising, Carrie Thearle, and Vic Wertz), and special thanks to Beth Moursund, Jim Northrup, Tom Jolly, Mons Johnson, David Sowell, Jennifer Schlickbernd, and Jesper Myrfors

THE ROBORALLY OPERATING MANUAL
TABLE OF CONTENTS

INTRODUCTION

It was a hard day on the planet. Widget manufacturing computer RSL973, or "Russel" for short, was a bit behind on his widget quota and the other computers had taunted him for it. He had hoped to get back at them in a game of RoboRally, but things weren't going well for him in that either.

Just this last turn, Russel's racing robot had been severely damaged and one of its program registers was stuck. According to the laws of the race, Russel would only get four instructions from the central race computer. Not much, Russel thought.

Russel's opponent in this race was control computer GNGR82, or "Ginger." Russel had nearly given up hope of winning this race. With this much damage, he could power down and sit idle for a turn, repairing all the damage, or he could try to make it to the nearest repair site. Either way, the odds of winning were mighty slim. But a quick calculation showed that fancy robotic footwork would win him some respect from the other planetary control computers and maybe they'd tease him less for missing his quotas.

The central race computer had finished its randomizing and started issuing the instructions that Russel and Ginger would use to program their racing robots. Ginger's robot was undamaged and received the full nine instructions. Ginger would have four instructions left over, while Russel would have to use all of his instructions.

And what poor instructions they were! Russel quickly realized there was no way he could program his robot to reach the repair site. But wait—what was this?

From deep in the back of Russel's "mind," his strategic subroutines suggested an alternate ploy: Russel had a slim chance of reaching the repair site this turn if Ginger could be coaxed into pushing him.

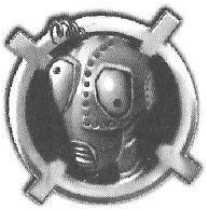
In a race, robots usually push each other as an offensive maneuver; a robot knocked off course might accidentally fall into a pit. That Russel was programming his robot to expect a push and use it to its advantage may at once keep him in the race and earn him the respect of his peer control computers.

Russel downloaded the instructions into his racing robot. He was back in the race!

GAME DESCRIPTION

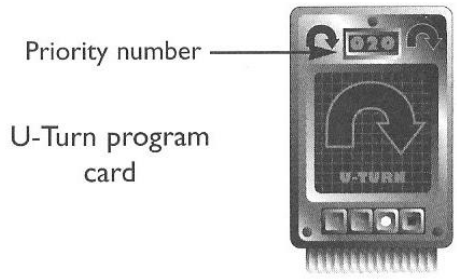
RoboRally is a robot race game in which each player attempts to be the first to touch a series of flags by maneuvering a robot across a dynamic race course. The game is for two to eight players playing independently or on teams. Frequently, the race will mutate into multi-player skirmishes as those players who are behind use their weaponry in an attempt to slow the opposition. The simultaneous movement rules encourage clever strategies and counter-strategies as players try to second-guess their opponents.

Good luck, and may all the conveyor belts be running your way!



COMPONENTS

- RoboRally Operating Manual
- 2 Factory Floor Guides—Quick-reference guides to the operation of all board elements.
- 6 factory tiles—Modular sections of the playing board. A factory tile is made up of squares, some empty and others containing board elements. Factory tiles are grouped together to make up the playing board. The six tiles that come with RoboRally are Cannery Row, Cross, Exchange, Island, Maelstrom, and Pit Maze.
- 84 program cards—Cards used to program how a robot will move. There are forty-two movement cards and forty-two rotate cards (18 x Move 1, 12 x Move 2, 6 x Move 3, 6 x Back-up, 18 x Rotate Right, 18 x Rotate Left and 6 x U-Turn). Each program card has a priority number (see sample below).



- 26 option cards—Extra equipment robots may acquire during the course of a race.
- 8 zinc robot miniatures (for "real" robots)
- 8 two-dimensional robot counters (for "virtual" robots)
- 6 flag counters
- 32 robot life tokens
- 61 damage chits

OVERVIEW OF GAME PLAY

This section provides an overview of how the game is played—the overview will hopefully make reading the rest of the rules easier and more enjoyable. Many of the game's key concepts are introduced in this section, so when the rules for these concepts are discussed later, you can already be familiar with the situations in which they occur. The second part of the operating manual, The RoboRally System Operators' Section, contains detailed rules that at least one of the players should know. While it isn't necessary for all the players to completely understand the System Operators' Section, it certainly helps.

RoboRally is a robot race game. Each of the players controls one or more robots that race against one another. The race course is marked by a set of flags spread out among the boards in play, and the object of the game is to have your robot be the first to touch all of these flags in order. Players are dealt a hand of program cards at the beginning of each turn, and these program cards allow a robot to move and rotate. Playing five program cards completes a turn, and after the five cards have been revealed and the actions on them performed, all the cards are collected, shuffled, and dealt out again.

Program cards aren't the only things to affect robot movement. Robots can collide, pushing each other off course, and active board elements like conveyor belts can also move robots. The board also has passive elements, such as pits (which will destroy robots) and walls (which will block robot movement).

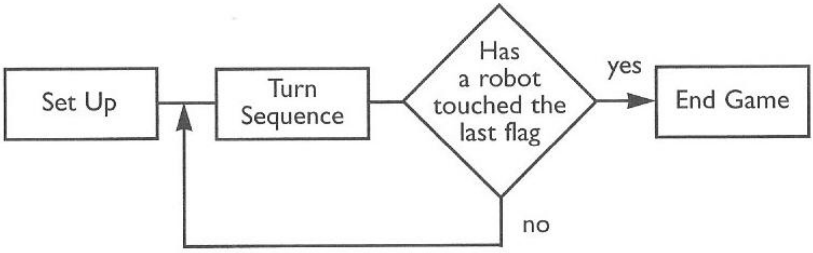
RoboRally is not just a good-natured race game. Laser beams crisscross the board, and lasers are also mounted on each robot. Robots hit by lasers take damage, and for each point of damage a

robot receives, the player is dealt one less program card each turn to program the robot. If a robot receives enough damage, some or all of the program cards will become "locked" and must be used again in the next turn. With even more damage, a robot is destroyed. Damage may be repaired by touching various repair sites on the game board or by choosing not to receive a set of program cards for a turn. A robot that doesn't receive program cards may still be moved by board elements, such as conveyor belts, and may still be pushed by other robots.

Each robot starts with three lives. Each time a robot is destroyed, it loses a life, but if the robot still has lives remaining it isn't permanently out of the game. A robot that is destroyed must start the race over at the last flag it touched, though that flag and any flags touched previously still count toward race victory. (So if a robot touched the first and second flags and was then destroyed, it robot would start over on the second flag and would only need to touch the third and fourth flags to win.) In addition to flags, several other places on the game board can serve as new starting positions. Players can start over at these places if their robot has touched them during the race.

In general, only one robot may occupy a square at a time. However, there are two occasions when more than one robot can occupy the same square: 1) at the beginning of the game, all robots start on the same square, and 2) if more than one robot reenters the race on the same turn, some of those robots may reenter on the same square. Both of these situations are handled by "virtual" robots. When a robot is virtual, it passes through other robots (without pushing them) and is immune to all effects caused by other robots, including lasers and pushing. However, virtual robots are affected by board-mounted lasers and every other board element. A virtual robot becomes real when it ends a turn in a square without another robot.

PLAY SEQUENCE



Set Up

To set up, choose a set of factory tiles that will make up the race track, lay the tiles out in the desired pattern, pick a starting position, pick the final and intermediate checkpoints (usually one checkpoint per tile), and place the robots on the starting position. A sample 2x2 race track is shown on the next page, and a section with more sample race tracks can be found on p. 44. (In all of the sample tracks, robots must touch the flags in order, but don't need to follow the indicated path.)

When you design your own course, choose one of the repair sites on the first board as the starting position. Also, the checkpoints should not be placed in a corner formed by two or more walls.

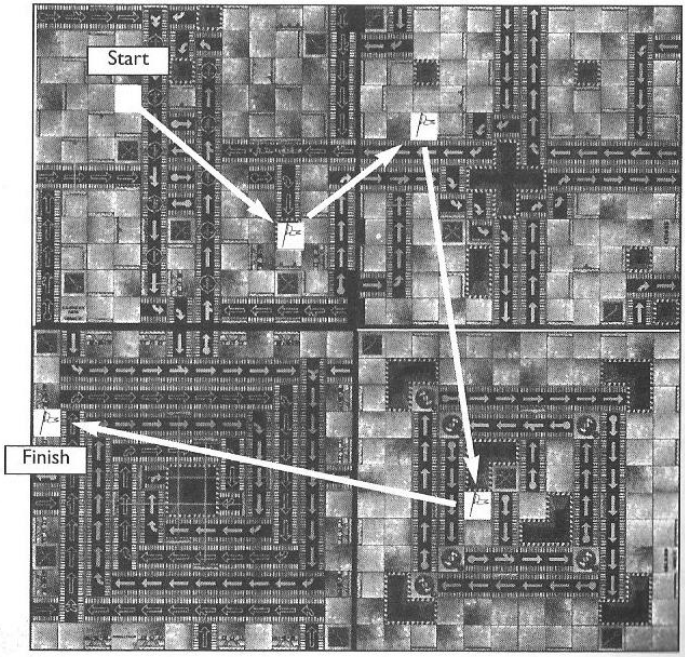
If you are using one of the sample race tracks, arrange the tiles and flags as shown. All robots start on the indicated starting square. Because all robots begin the game as virtual robots, use the two-dimensional robot counters (for more information, see **Virtual Robots** on p. 15). Players may choose the direction their robots will initially face. Finally, players take two tokens apiece to represent their robot's "extra" lives. (Each robot has a total of three lives.)

Optionally, the following rules may be used to determine the number of starting lives:

Start with 3 lives, then
- add 1 life for 5 or more players
- add 1 life for 5 or more factory tiles

So if you have five players on 6 tiles, each player would start with 5 lives (a base of 3, plus 1 for the number of players, plus 1 for the number of tiles).

As many as eight people can play RoboRally, but first-time players will find it easier to learn with four players or less. RoboRally also makes an enjoyable solitaire game, so why not try it now right out of the box?



Turn Sequence Overview

Each turn begins by reshuffling the cards from the previous turn and then dealing a hand of program cards to each player. The programs that players choose from these cards are then played out over five "register phases." After the end of the fifth register phase, a few end-of-turn actions (which will be described later) are resolved, and the turn ends.

At the start of each turn, players are dealt nine cards from the deck of program cards. The cards that a player selects from this hand will determine how a robot will move across the factory floor. The deck consists of cards like Move 2, Rotate Right, Rotate Left, Back-Up, and others. Players select five of the nine cards and lay them out face down in the order they want the programs executed. Players simultaneously reveal their first program card and move their robots according to the programs on each card. The effects of simultaneous movement and board elements are then resolved. After every robot has been moved, the next program card is revealed. Since the cards are revealed in a particular sequence, the order in which they are arranged is important. Players should lay out their program cards in a way that clearly shows in what order they are to be executed. The first card is said to be in the first "register," and it is revealed during the first register phase.

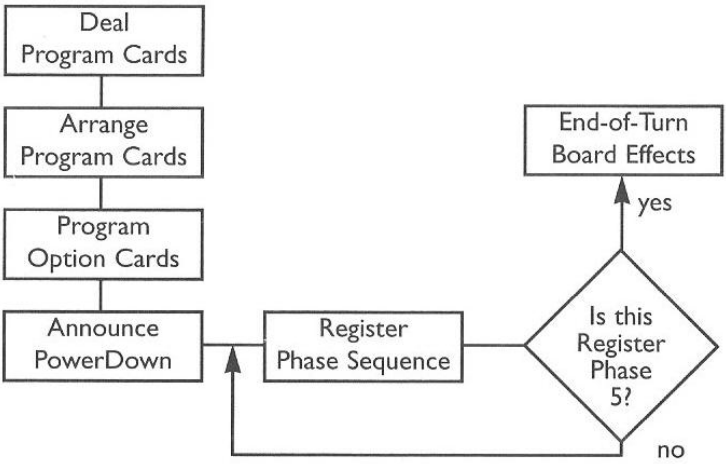
Has a Robot Touched the Last Flag?

The turn sequence is repeated until a robot has successfully touched all the flags in order. To touch a flag, a robot must end a register phase on the same square as the flag.

End Game

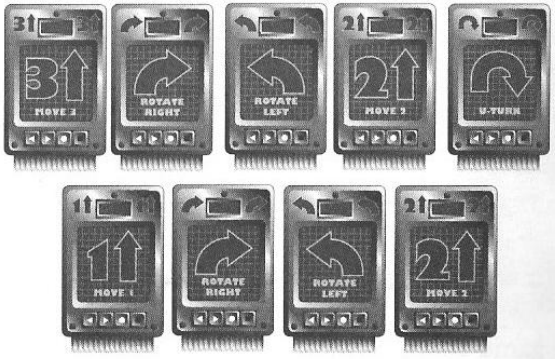
The winner is the first to touch all the flags in the indicated order. The game can end when the winner has touched the last flag, or play can continue to determine runners-up.

TURN SEQUENCE



Deal Program Cards

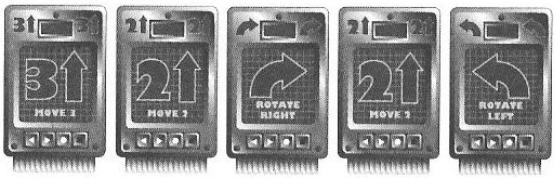
Shuffle the program card deck and deal each player with an undamaged robot nine cards. Players with damaged robots receive fewer cards. A robot with 1 point of damage receives eight program cards, a robot with 2 points of damage receives seven, etc. If a robot has more than 4 points of damage, some registers will be "locked" (for more information, see **Locked Registers** on p. 26). A typical hand is shown below.



Any robots that were destroyed on the previous turn now reenter the game at their last "archive" location. If two or more robots reenter on the same archive, then these robots start the turn as virtual robots (for more information, see **Withdrawing an Archive Copy** on p. 30).

Arrange Program Cards

Players now each choose the five program cards from their hand that they wish to use and arrange them face down in the order in which they will be executed, discarding any remaining cards. For example, if you were dealt a Move 3, two Move 2s, a Move 1, two Rotate Rights, two Rotate Lefts, and a U-Turn (as shown on the previous page) you might choose to arrange your cards like this:



You can indicate this by either placing the cards face down from left to right or by stacking them with the first one on top.

Program Option Cards

Some players may acquire option cards in the course of the game, and some of these cards require programming. An example of an option card that must be programmed is Shield, which protects one side of a robot from damage. Shield is programmed by indicating at this time which side of the robot is to be protected.

Announce PowerDown

A robot can use a PowerDown to repair damage. Only damaged robots may choose to power down. When a robot powers down, all damage is immediately repaired. Any damage sustained while the robot is powered down is, unfortunately, new damage that must be repaired in a subsequent PowerDown or in another manner. A PowerDown announced this turn takes effect on the next turn (for more information, see **PowerDown** on p. 26).

Register Phase Sequence Overview

In every "register phase," each player reveals a program card and then any subsequent activities are resolved. During a register phase, a robot moves according to its program card for that phase. After this initial movement, certain board elements may affect robots. (For example, a robot on a gear would be rotated 90°.) After board elements move, laser fire is resolved, and finally the end-of-register-phase effects are resolved. Each of these actions is described in more detail in the **Register Phase Sequence** section starting on page 16.

Is This the Fifth Register Phase?

Each turn consists of five register phases, one for each program card. After the fifth register phase is executed, the turn sequence is nearly over.

End-of-Turn Board Effects

Certain activities take place only after the fifth register phase has been completed. Robots on a repair site or checkpoint may now repair 1 point of damage, and robots on a two-wrench repair site may either repair 2 points of damage or receive an option card. Players whose robots were powered down this turn must now decide whether their robots will remain powered down or

"power up" and receive cards in the next turn. Also, robots that were virtual this turn and are now occupying a square without other robots are turned into "real" robots. After these actions have been performed, it is time to begin the next turn.

Virtual Robots

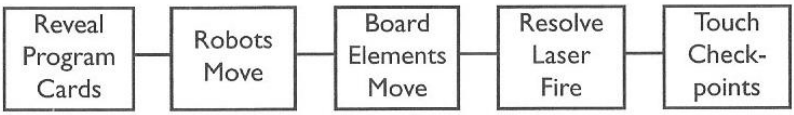
Situations arise in the game when two or more robots occupy the same square. At the beginning of the game all robots start out on the same square, and two or more robots may reenter the game on the same square after dying the previous turn. Robots in either of these situations enter the race as virtual robots.

A virtual robot doesn't interact with other robots. It doesn't shoot other robots, and other robots don't shoot it; virtual robots aren't pushed by other robots, and other robots don't push virtual robots. However, virtual robots are affected by all board elements; they are blocked by walls, can fall into pits, can be shot by board-mounted lasers, and are affected by conveyor belts, gears, etc. (Until you become real, you can pretend your robot is the only one on the board.)

At the end of a turn, if a virtual robot is on a square all by itself, it then becomes a "real" robot. When this occurs, the two-dimensional robot counter should be replaced by its three-dimensional counterpart.

REGISTER PHASE SEQUENCE

This section describes a single register phase sequence. This same sequence is repeated five times during a turn.



Reveal Program Cards

Each player's program card is revealed simultaneously.

Robots Move

Robots move as their cards indicate. A Move 2 will move a robot forward two squares, a Back-Up card will move a robot back one square, and so on. Robots executing rotate cards turn 90° in the indicated direction, and a U-Turn is a 180° turn.

The tiny numbers on the top of each program card indicate a robot's priority for that register phase. For the most part, robots may be moved simultaneously, however occasions often arise when robots are close together and the order in which robots are moved becomes critical. In these cases, robots are moved according to the number on the program card. A higher number means a higher priority, so a robot with a 200 would move before a robot with a 100.

Board Elements Move

After all robots have executed their program card for the phase, the board elements take effect. Pushers will push a robot, gears will rotate a robot, and conveyor belts will move a robot. In some cases, more than one board element may affect a robot in a reg-

ister phase. A complete listing of all the board elements, including what they do and when they do it, can be found on the Factory Floor Guide.

Note that there are several kinds of board elements. Some perform their operations all the time, like walls and pits, and some perform their function only after robots have moved, like gears and conveyor belts.

Resolve Laser Fire

Laser fire is the primary way robots get damaged in RoboRally. Robots that stop in a square with laser beams going through it take a point of damage for each beam in that square. A laser beam will not pass through a robot, so if two or more robots are in the same beam, only the one closest to the source of the laser is damaged.

Also, every robot has a forward-firing laser. A robot in another robot's line of sight (directly in front of it with no intervening obstruction, like a wall or another robot) will automatically take a point of damage from that robot's forward-firing laser.

Finally, note that the sequence of events is critical. A robot can move through a laser beam undamaged—the laser only damages the robot if it remains in the beam after it has moved and after all board elements have moved.

Touch Checkpoints

Any robot that is still alive at this time and is on a checkpoint may now consider it "touched." Robots on checkpoints or repair sites have now updated their archive location and, as a result, will reenter the race at that checkpoint or repair site should they meet their demise before reaching another. Robots that have touched a check-

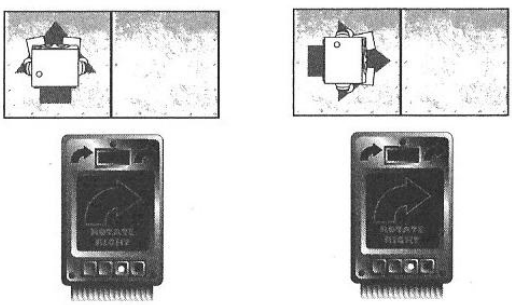
point can now move on to the next one in sequence. The sequence of events is critical here, too. A robot could move onto a checkpoint during the Robots Move segment, but be fired upon during the Resolve Laser Fire segment and receive enough damage to destroy it. In this case, the robot would not have touched the checkpoint.

This concludes a single register phase sequence. This sequence is repeated five times during the course of a turn.

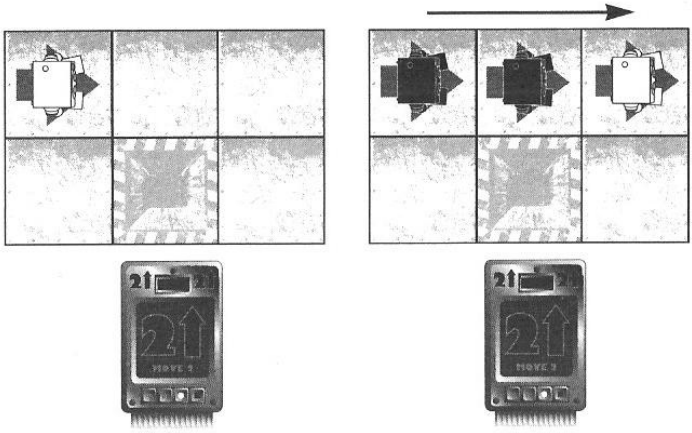
Examples

The following figures show examples of robot movement. Robots may go through several parts of this sequence in one panel. In these cases, each step in the sequence shows the robot in black, and the final step in the sequence shows the robot in white.

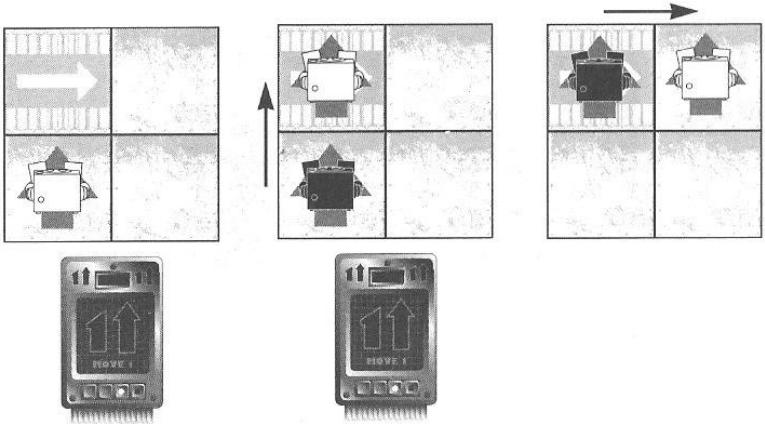
A Rotate Right program card, when revealed, turns the robot 90° to the right. The card rotates the robot, but doesn't move it.



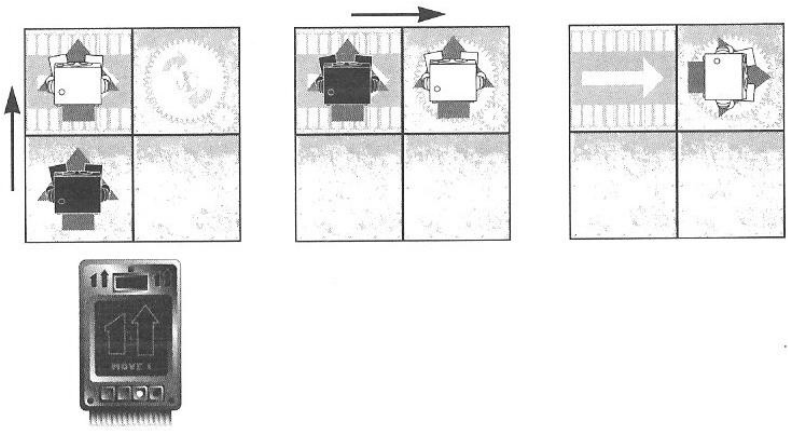
A Move 2 program card moves the robot forward 2 squares. The robot moves through the intervening square. A robot in an intervening square would not prevent the robot from executing the movement but instead would be pushed forward. Walls and pits will prevent a robot from moving its full motion; a wall will simply stop a robot without damaging it, but a pit will destroy any robot that moves over it.



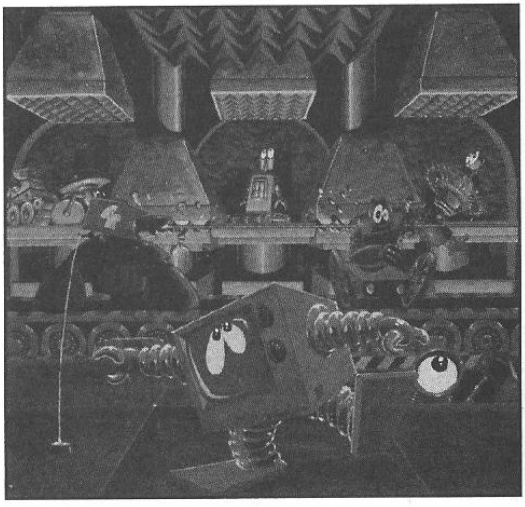
This is another example of a movement card. This time it is a Move 1 card, but it moves the robot onto a conveyor belt. In the register phase sequence, first the robot moves, then the conveyor belt moves.



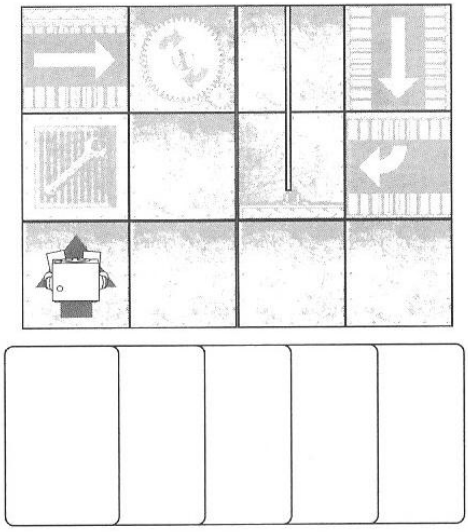
This is another example of a movement card. This time there is a Move 1 card and both a conveyor belt and a gear. The Factory Floor Guide states that first conveyor belts move, then pushers, and then gears.



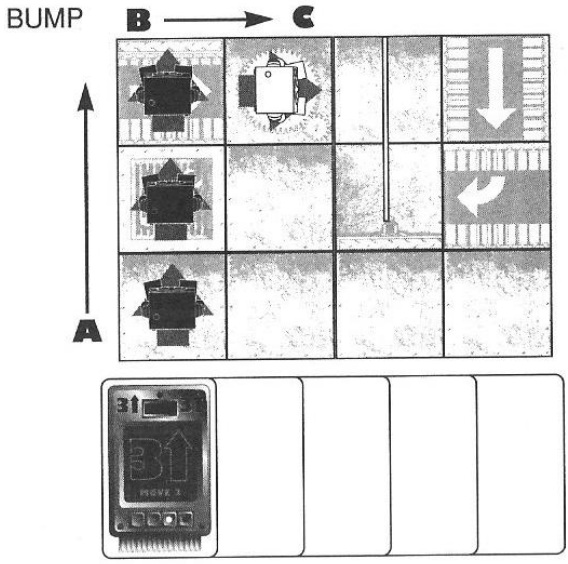
So first the robot moves onto the conveyor belt by executing the Move 1 card. Then the robot is moved onto the gear by the conveyor belt. The gear then rotates the robot 90° to the right. This situation of having more than one board element affect a robot in a single register phase is quite common.



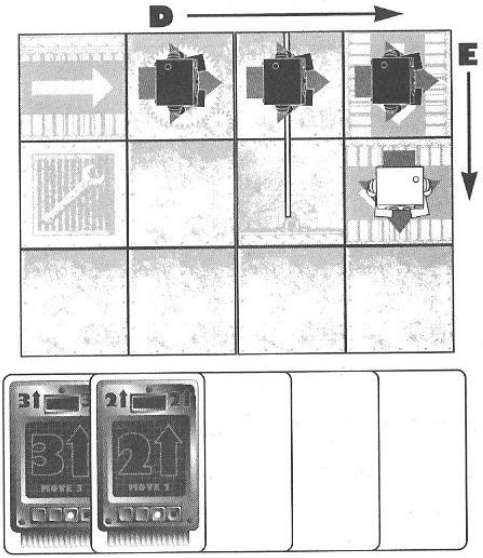
A COMPLETE SAMPLE TURN



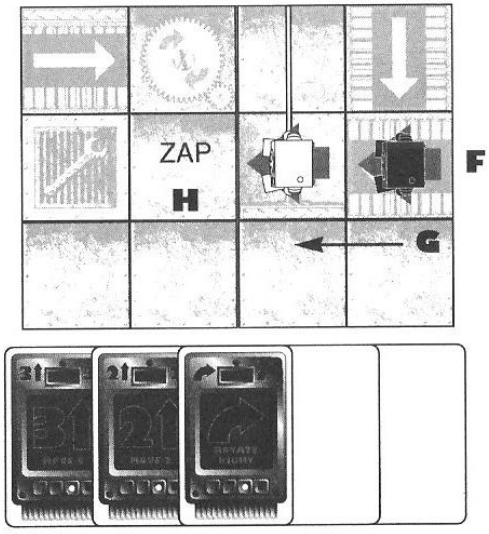
First the robot starts the turn in the square indicated above. On the first register phase, the robot executes a Move 3 program card as shown on the next page (A). This takes it over the wrench and causes it to bump into the wall. The robot did not stop on the wrench, so it didn't "touch" it, and the wall interrupted the Move 3 card, effectively making it a Move 2 card. This completes the robot's movement in the first register phase. Next comes board element movement. As shown in the example on the previous page, the conveyor belt moves first (B), moving the robot 1 square onto the gear. The gear then moves (C), rotating the robot 90° to the right. This ends the first register phase.



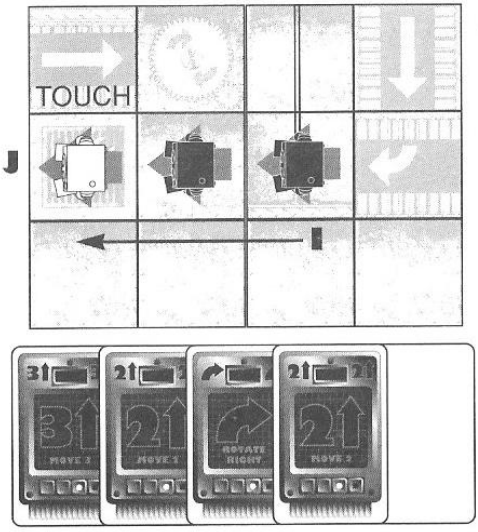
On the second register phase, shown below, a Move 2 card is revealed. This takes it through the path of the laser (D), and it comes to rest on the conveyor belt. The robot did not stop in the laser beam, so it is unharmed. Next the board elements are active. In this case, since the conveyor belt moved the robot onto a turning conveyor belt, the robot is both moved and rotated simultaneously (E). This ends the second register phase.



On the third register phase, a Rotate Right card is revealed. The robot is rotated 90° to the right (F). Then, since the robot is still on a conveyor belt, it is pushed one square into the laser beam (G). Finally, since all the board elements have performed their actions and this robot is still in the path of a laser beam, it is hit by the laser and suffers 1 point of damage (H). This ends the third register phase.

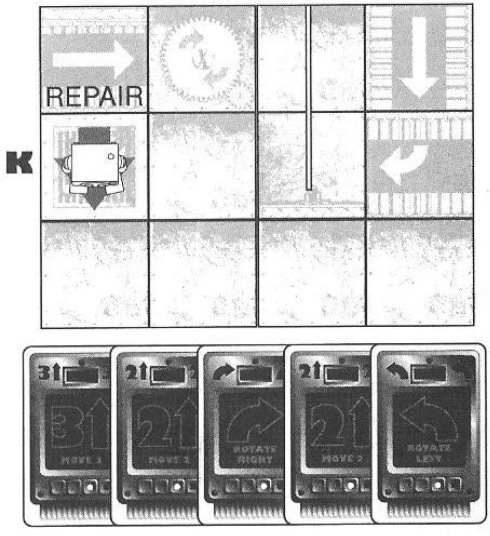


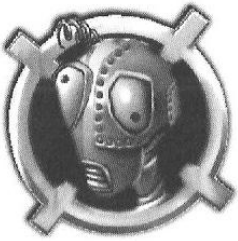
On the fourth register phase, a Move 2 card is revealed, as shown on the next page. The robot moves two squares (I). This places it on the wrench. The robot comes to rest there, since there are no conveyor belts or other robots about to push it off. It has now touched the wrench (J). Touching it means that the next time this robot is destroyed, it will be rebuilt on this location. This ends the fourth register phase.



Damage Counter

On the fifth register phase, a Rotate Left card is revealed. This rotates the robot 90° to the left (K). This action ends the fifth register phase, and, because the robot is still on the one-wrench repair site at the end of the fifth register phase, it gets to repair 1 point of damage. This ends the turn.





The RoboRally System Operators Section

DAMAGE

Robots receive damage from lasers, and archive copies of robots start with 2 points of damage. Each time a robot receives a point of damage, the player takes a damage chit.

Any damage received decreases a robot's "intelligence," meaning that damaged robots get fewer program cards than undamaged robots. For each point of damage a robot receives, the player is dealt one less program card.

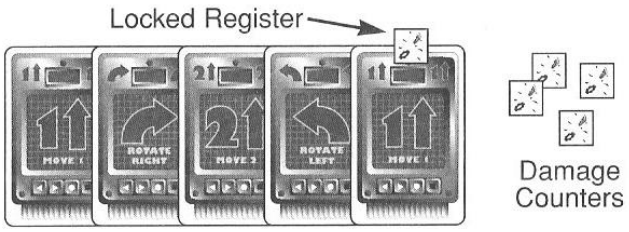
LOCKED REGISTERS

Since undamaged robots use nine program cards, when a robot has 9 points of damage it won't receive any new program cards. It may still move, even with 9 points of damage, because all its registers will be "locked"—this means that the program cards from the previous turn are not discarded, and the previous turn's program is executed again. The table below summarizes the cumulative effect of damage to a robot.

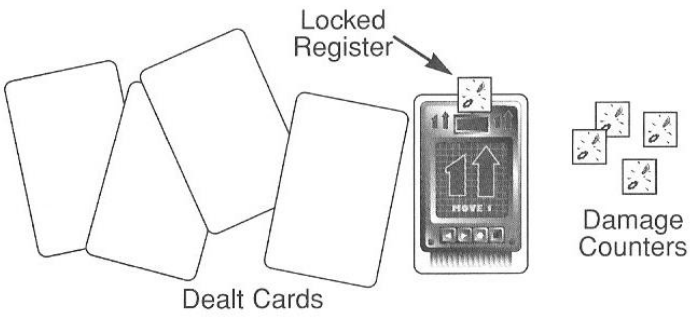
DAMAGE	EFFECT
0	receive 9 program cards
1	receive 8 program cards
2	receive 7 program cards
3	receive 6 program cards
4	receive 5 program cards
5	receive 4 program cards, lock the fifth register
6	receive 3 program cards, lock registers 4 and 5
7	receive 2 program cards, lock registers 3,4, and 5
8	receive 1 program card, lock registers 2,3,4, and 5
9	receive no program cards, lock all registers
10	destruction

Each time a robot receives a point of damage that robot receives a damage chit. Players mark a locked register by placing the chit on top of the card in that register. Remember that a locked register is a side effect of damage—registers can only become locked when a robot receives damage.

Below is the program of a robot that has just received its fifth point of damage. One of the damage chits is partially covering the card in the fifth register, signifying that the register is locked.



At the end of this turn, four of the five program cards are returned to the program deck, and only four are dealt back to this player. The fifth card remains in the fifth register as shown below.



Locked registers become unlocked (and the program card is discarded) when the damage locking it is repaired.

REPAIRING DAMAGE

Robots repair damage by using repair sites or by powering down. Also, an "option exchange" will allow a robot to avoid receiving a point of damage.

Repair Sites

One way to repair damage is to end a turn on a repair site. Repair sites are denoted by wrenches; sites with a single wrench repair 1 point of damage and sites with two wrenches repair 2 points. Damage doesn't have to be repaired in the same order it was received; for instance, if registers 3, 4, and 5 are locked, a player can choose to repair the point of damage locking the fourth register. (Also, a player ending a turn on a two-wrench repair site may choose to take an option card instead of repairing 2 points of damage.)

PowerDown

The other method for repairing damage is to power down. A robot that is powered down does not receive program cards and all damage to that robot is repaired.

If a player wants her robot to power down, she must announce this a turn in advance, immediately after cards for the current turn are dealt and programmed. Her robot must play out the current turn and the robot powers down on the next turn. At the beginning of the next turn, all damage chits are discarded and the player receives no program cards.

Example

It's the fifth turn, and Joey's robot has just been fired upon four times this turn. He has 4 points of damage, and he would like to power down as soon as possible. At the beginning of the sixth turn he gets five program cards and arranges them. Then, after everyone has arranged their cards, Joey announces that he will power down on the seventh turn. The seventh turn comes, Joey gets no cards, and his 4 points of damage are immediately repaired.

Before cards are dealt for the next turn, a player may decide to leave her robot powered down. At the beginning of each turn

that a robot is powered down, all damage is discarded. Immediately after withdrawing an archive copy, robots may choose to start out powered down.

Example

During the course of the seventh turn, Joey gets shot twice (while powered down!) by other robots. Before cards are dealt for the eighth turn, he announces his decision to stay powered down. As the other players are dealt cards for the eighth turn, Joey discards his two new damage chits.

A powered-down robot is completely shut down—it can't fire weapons, tag checkpoints, or update its archive location, nor can it acquire or use option cards. Powered-down robots don't move under their own power (they receive no program cards), but they may be moved by pushers, gears, and conveyor belts. They may also be pushed by other robots and shot at by lasers.

Option Exchange

A robot with option cards may choose to sacrifice an option card to avoid receiving a point of damage. (The option takes the damage instead of the robot.) Any number of option cards may be discarded, but the player must decide to do an option exchange at the time the damage is received.

DESTRUCTION

A robot is destroyed when it receives its tenth point of damage, falls into a pit, moves off a blank edge of the board, or is crushed by a crusher. A destroyed robot immediately loses an option card of the player's choice. If this is the third time that a robot has been destroyed, that robot is permanently out of the

game. Otherwise, the robot may reenter the race by withdrawing an archive copy.

WITHDRAWING AN ARCHIVE COPY

A robot that has been destroyed must have touched an archive location at sometime during the race (every robot starts out on one). A robot will reenter the race at the last archive location (either a checkpoint or a repair site) that it touched. This is called "withdrawing an archive copy."

A player withdrawing an archive copy places his robot on the last archive location touched; two damage chits are taken, and the player chooses the direction the robot will face.

If another robot is already on that square, the player withdrawing the archive copy starts with a virtual robot. If another player is also withdrawing an archive copy on the same checkpoint or repair site, both players start out with virtual robots. Otherwise, players start out with "real" robots.

Before program cards are dealt, a player may decide to reenter the race powered down (to repair the 2 damage points). It is possible to reenter as a powered-down virtual robot!

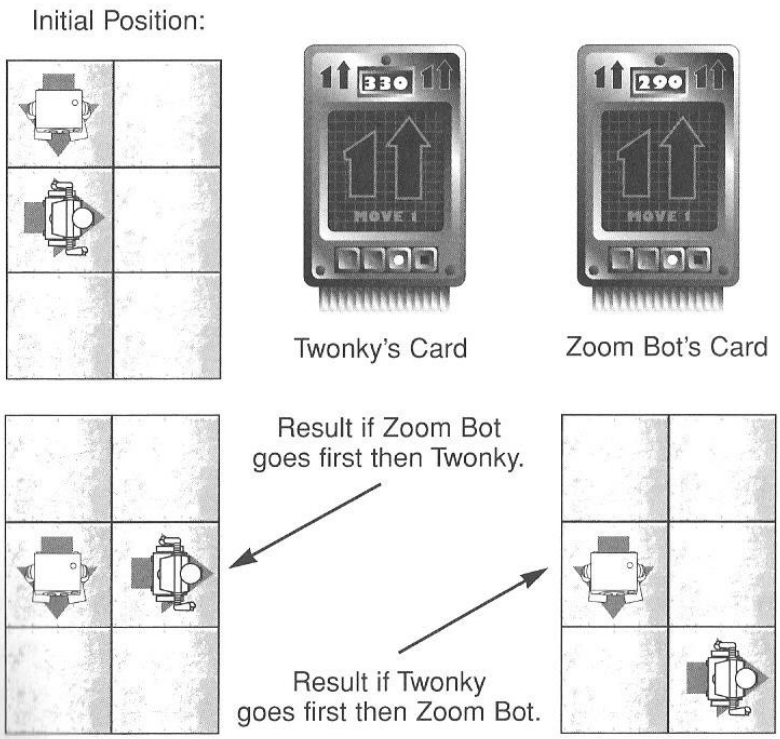
CARD PRIORITY

Robot movement isn't strictly simultaneous; however, most of the time it can be considered simultaneous, and moving robots simultaneously will speed play. Players should pay attention, though, to card priority when two robots attempt to move into the same square, or when one robot leaves a square while another enters it.

As a general rule, movement cards have a higher priority than rotate cards. The faster movement cards are executed before the slower movement cards, and the rotate cards are executed last. In addition, each program card has a priority number on it; higher numbers move before lower ones (card number 200 moves before card number 100, and so on).

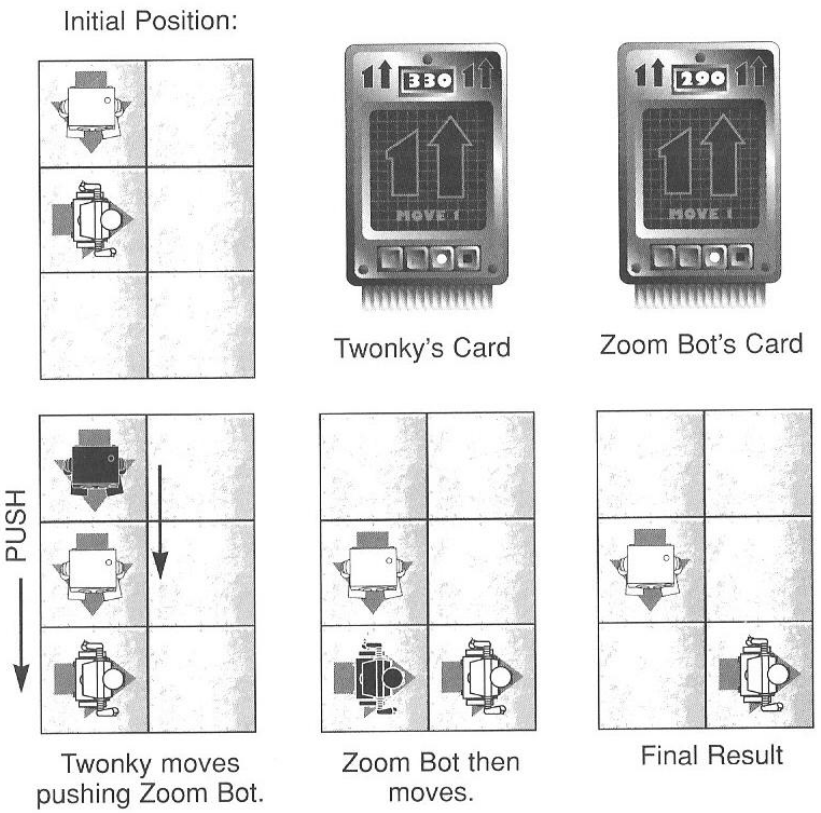
ROBOT PUSHING

Situations where the card priority must be followed usually result in one robot pushing another. Consider the figure below:

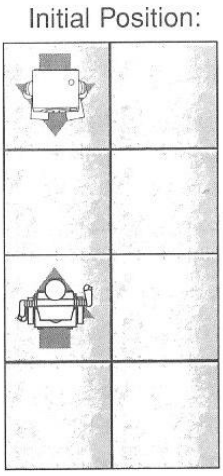


In this case it matters which robot moves first. In one instance the robots collide and one pushes the other, and in the other instance they don't hit at all.

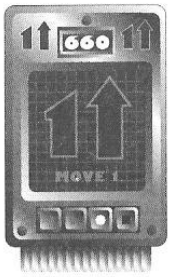
When robots collide, one will push the other. The robot that moves first is the one with the highest priority number. In the example below, Twonky has a priority number of 330 while the Zoom Bot has a priority number of 290. Twonky moves first, pushing the Zoom Bot as shown below.



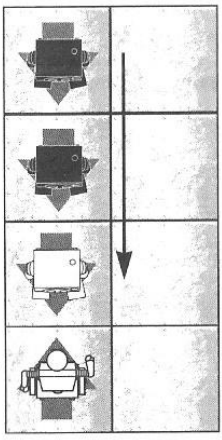
On the next page is another example of robots pushing each other. This time the numbers on the cards don't need to be compared since the Move 2 card will move before the Move 1 card. (The Move 2 card, though, does have a higher priority number.)



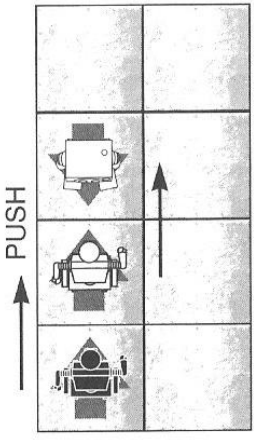
Twonky's Card



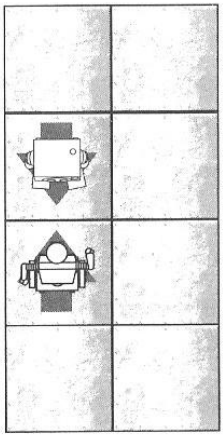
Zoom Bot's Card



Twonky moves pushing Zoom Bot.



Zoom Bot then moves pushing Twonky.

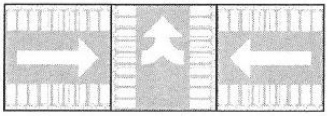


Final Result

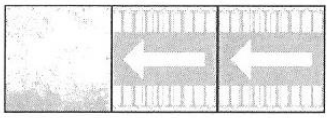
CONVEYOR BELT PRIORITY

Normally when conveyor belts move, all robots on a conveyor belt are moved simultaneously. One robot can be directly behind another, and if both robots are on a conveyor belt when it's time for conveyor belts to move, both will be moved.

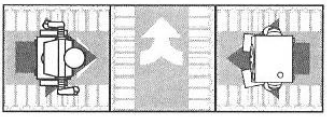
However, there are several situations in which more than one conveyor belt will converge onto the same square.



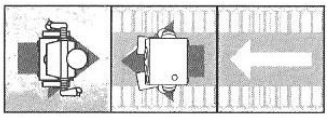
Conveyor belt position



Conveyor belt position



Conveyor belts do not move either robot.



Conveyor belt does not move Twonky.

In the example on the left, the belts are attempting to push both robots into the same square. If this happens while conveyor belts are moving, neither robot moves. However, if the robots had been moving under their own power (during the robot movement segment of the register phase sequence), one robot would have had a higher priority number than the other and would have moved first. Here, though, robots moving under the influence of conveyor belts never push other robots. (If the situation is ambiguous, don't move either robot.)

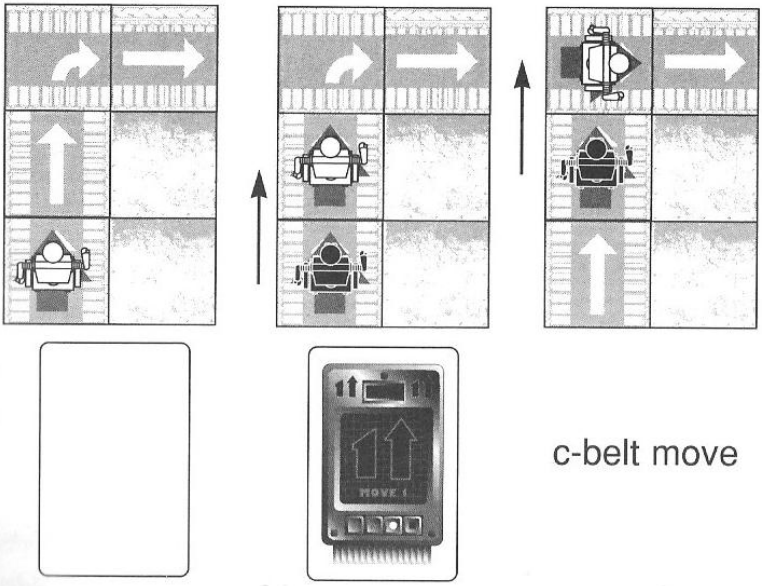
Robots moving under the action of conveyor belts never push other robots; this rule also applies to the example on the right. In this situation, a robot has finished its move on an empty square that has a conveyor belt emptying onto it, and another robot is on that conveyor belt. When it's time for conveyor belts to move, the robot on the belt is blocked by the robot on the empty square.

TURNING CONVEYOR BELTS

Turning conveyor belts turn robots only if the robot is moved onto one by another conveyor belt. This is true even when a

robot is moved from an express conveyor belt onto a turning, normal belt. If the robot is moved onto a turning conveyor belt by any other means (either under its own power or by being pushed), the belt does not turn the robot, but simply pushes it forward.

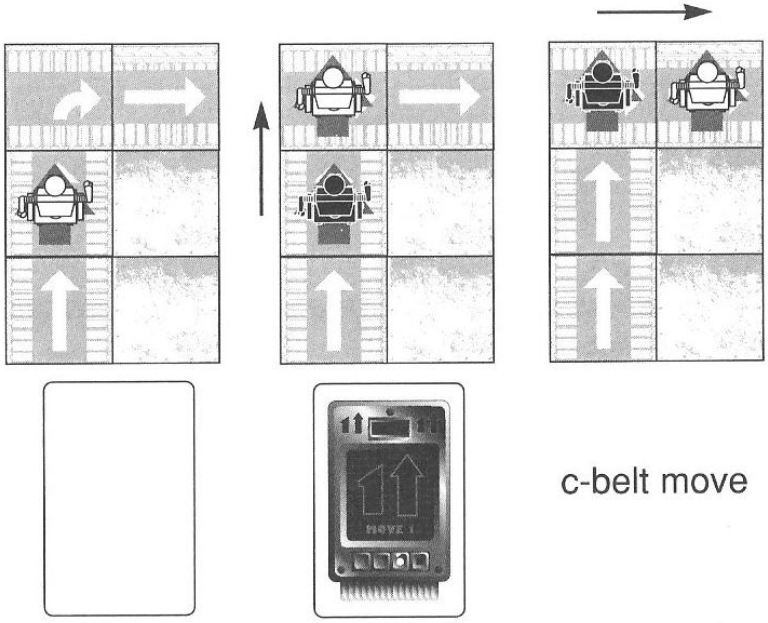
Here are some examples:



c-belt move

The robot shown above moves forward one square under its own power. The conveyor belt then moves and pushes it onto a turning conveyor belt. Because the robot was moved onto the turning conveyor belt by another belt, it was rotated 90° as shown.

This differs from the example below. In this example, the robot moves onto the turning conveyor belt under its own power. Then the turning conveyor belt moves and pushes the robot onto the next square without rotating it. (Even if the robot had been pushed onto the belt by another robot, it still wouldn't have been rotated.)



c-belt move

OPTION CARDS

At a two-wrench repair site, robots may receive an option card instead of having 2 points of damage repaired. Some option cards are extra weapons like Rear Laser; some are weapon enhancements, like Double Barrel Laser or Fire Control; some are simply weapon-

related, like Shield or Ablative Coat; and some are racing-related, like Recompile or Fourth Gear. Unlike the priority numbers on program cards, the catalog numbers on option cards have no effect on game play. To assist in understanding options, some general rules apply.

Optional Weapon

An optional weapon may be used anytime the main laser may be fired. (However, either the main laser or an optional weapon must be used.)

Example

Tractor Beam is an example of an optional weapon. If a robot with a Tractor Beam has an opportunity to shoot another robot, it may do so with either the Tractor Beam or the main laser. However, it must choose one, and it must fire.

Main Laser Mod

A main laser modification is an enhancement to the main laser and is permanently active. Players may not choose to use a main laser mod like they can optional weapons. Main lasers mods must be used when the main laser is used.

Example

Double Barrel Laser is an example of a main laser mod. If a robot with a Double Barrel Laser shoots another robot with its main laser, it must do so with the Double Barrel Laser, dealing 2 points of damage. (The main laser has been modified to always deal 2 points of damage, and the robot may not turn off the Double Barrel Laser and just deal 1 point of damage.)

Additional Weapon

Additional weapons are weapons that are used in addition to the main laser. Players must always use an additional weapon when appropriate.

Example

Rear Laser is an example of an additional weapon. If a robot with a Rear Laser has an opportunity to fire at a robot with its main laser (if a robot is directly in front of it) and with its Rear Laser (if a robot is directly behind it), then it must fire at both robots.

Turn Programmed

An option card that is turn programmed is programmed during the Program Option Cards segment of the turn sequence. How to program an option card is usually straightforward and written on the card.

Example

Shield is an example of a turn programmed option. It protects one side of a robot from laser fire. It is programmed by indicating each turn which side the Shield will protect.

Run Time

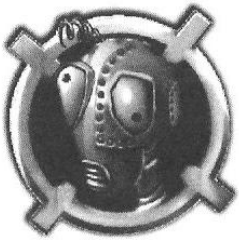
Run time options can be used anytime the option card indicates it is appropriate. They do not need to be programmed in advance.

Example

Fourth Gear is an example of a run time option. Fourth Gear allows a robot to move forward four squares anytime it is executing a Move 3. "Run Time" here means that the player may choose to use the option as she's using the Move 3 card. She doesn't have to decide to use it when she's programming her hand, and she may choose not to use the option at all.

Other

Other options have detailed instructions about their use on the cards themselves. Remember that, in general, any option may be exchanged to prevent a point of damage as indicated in the **Repairing Damage** section.



The RoboRally Glossary

Archive Copy: *When a robot is destroyed, a new copy is quickly assembled and materialized onto the factory floor at the last archive location for that robot. Due to the haste of construction, archive copies start with 2 points of damage.*

Archive Location: *Each time a robot touches a* **checkpoint** *or* **repair site**, *that spot becomes its archive location. If a robot is destroyed, an* **archive copy** *of that robot is created at its archive location.*

Board: *one of the game boards that make up the Factory Floor. Also called a* **factory tile**.

Checkpoint: *a flag which must be touched as part of the race course. Only checkpoints touched in the correct order count. Each checkpoint also serves as a one-wrench* **repair site** *and an* **archive location.**

Conveyor Belt: *rapidly moving belts which carry materials from one part of the factory to another, as well as carrying waste materials to disposal areas. Robots which stray onto these belts will be carried along as well.*

Crusher: *hydraulic presses which flatten any robot careless enough to wander underneath at the wrong time.*

Damage: *the factory robots, while dimwitted, are quite sturdily constructed, and can continue to operate even after being damaged. Each laser that hits a robot does one point of damage; to keep track of damage, take one* **damage chit** *per hit. See also* **locked register** *and* **option card**.

Damage Chit: *the counters used to record damage to robots.*

Destruction: *a robot is destroyed when it falls into a* **pit**, *is flattened by a* **crusher**, *or accumulates ten* **damage chits**.

Factory: *the computer-controlled complex where RoboRally is played.*

Factory Floor Guide: *a handy reference sheet describing the operation of all the various devices on the factory floor.*

Factory Tile: *one of the segments that makes up the Factory Floor. Also called a* **board**.

Flag: *bright-colored pennants marking the* **checkpoints** *that form a RoboRally race course. Robots must touch each flag in sequence in order to win.*

Gear: *a rotating device on the Factory Floor. Also, the speed of a robot, i.e. first gear, second gear, third gear, and reverse.*

Laser Beam: *industrial lasers powerful enough to drill a tidy hole through any robot stopping in their path. Numerous lasers are mounted on the factory floor, and each robot is also equipped with one.*

Laser Mount: *the device that projects a* ***laser beam****.*

Life Token: *counters used to keep track of how many* ***archive copies*** *a robot has used. Each robot is allowed 4 copies in addition to the original before being permanently junked.*

Line of sight: *lasers and other weapons are restricted to firing in a straight line along a column or row. Line of sight is blocked by the first intervening wall or robot. Diagonal firing is not possible.*

Locked Register: *when a robot accumulates five or more points of damage, its brain begins to lose functionality, causing it to repeat part of its previous programming instead of accepting new orders.* ***Damage chits*** *after the fourth are placed on top of the robot's program cards, starting with the last one, marking these registers as locked and unalterable.*

Movement Card: *a* ***program card*** *instructing the robot to move at a particular speed, either forward or backward.*

Operating Manual: *the RoboRally rulebook.*

Option Card: *these cards represent various pieces of equipment which can be added to robots. When a robot is damaged, you may choose to discard an option card instead of taking a* ***damage chit****.*

Pit: *a deep hole in the floor of the factory, leading to a garbage crusher, furnace, or other pleasant destination. Any robot falling down a pit is doomed.*

PowerDown: *a valuable robot ability, which allows them to shut down and go into self-repair mode, losing a turn and repairing all damage.*

Program: *the robots used in the widget factory are made by the lowest bidder and have very limited programming capabilities. A robot's program consists of five* ***program cards*** *placed in the order that they will be executed.*

Program Card: *used to tell the robots what to do, each program card describes one specific movement or rotation. A program card can either be a* ***movement card*** *or a* ***rotate card****.*

Programming: *the process of picking out program cards to make your robot go where you want it to, or at least as close as it possibly can (preferably without falling into a pit first). See* ***robot dance****.*

Pusher: *immense, rubberized wall-sections which are used to move parts short distances when conveyor belts are inconvenient or timing is critical. Robots standing in front of one of these when it pulses will be harmlessly pushed out of the way. (But what they're pushed into may not be so harmless.)*

Register: *the on-board memory of the factory floor robots is quite limited, consisting of only five registers. Each of these registers can hold one* ***program card*** *and instruct the robot to do one thing.*

Register Phase: *part of the turn, consisting of revealing one* ***program card****, moving the robot, applying the effects of the various devices on the factory floor, resolving laser fire, and applying end-of-phase effects.*

Repair Site: *a spot on the factory floor marked with one or two wrenches. Any robot ending a turn on one of these sites gets a quick lube job, removing one damage chit per wrench. The two-wrench spots can add an* ***option card*** *to the robot instead of repairing damage. Each* ***checkpoint*** *also functions as a one-wrench repair site. All repair sites also serve as* ***archive locations****.*

Robot: *those little guys who get shot, smashed, and generally mistreated in the course of the game.*

Robot Dance: *the twitching, turning motions RoboRally players tend to make as they look between their program cards and the board, trying to figure out how to get a robot from "here" to "there," and is that a left or a right turn?*

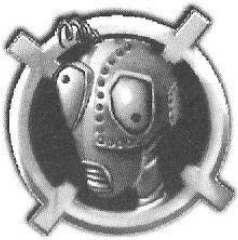
Rotate Card: *a* ***program card*** *instructing the robot to rotate left, right, or all the way around. A robot cannot move and rotate at the same time.*

Touch (a checkpoint or archive location): *a robot is considered to have touched a* ***checkpoint*** *if it ends a register phase there. Driving over a checkpoint, or driving onto one and being shoved away by another robot, does not count as touching. Neither does stopping on a checkpoint but being destroyed by laser fire.*

Turn: *each turn in RoboRally consists of dealing out* ***program cards****, programming your robot, and playing out five* ***register phases****.*

Virtual Robot: *when two robots start the turn on the same spot, they are unable to fully materialize and are referred to as virtual. A virtual robot is phased in enough to interact with the factory floor devices, including lasers, but not with other robots or robot lasers. A virtual robot materializes the rest of the way when it ends a turn in a spot that doesn't contain any other robots.*

Wall: *sturdy, solid walls that separate parts of the factory from each other or provide a place to attach equipment. Robots running into walls take no damage (they have rubber bumpers), but the robots don't go anywhere. Lasers can't shoot through walls.*



Sample Race Tracks

Game Sequence Diagram

Robot Roll Call

RACE TRACKS

This section contains several sample race tracks, including tracks appropriate for beginning players. In addition to sample tracks, this section also provides guidelines and suggestions for race track creation; after you've played the game a few times, you will develop a sense of where to place flags and how to put together your own race tracks.

Guide to the Courses

Each course has a suggested number of players, an estimated course complexity, and an approximate playing time. The time and complexity are largely based on the number of players, so fewer players will usually make for an easier (and shorter) game, while adding players increases complexity and play time.

Complexity is a rough measure of how difficult the programs will be to create and how often robots will interact. High player interaction means programs must take into account not just the board, but other robots as well.

Time is the typical time that moderately experienced players will take to play the game. (Beginners should expect to take longer.) If a track is rated as short, play time is about an hour; a moderate track takes about one to two hours, and a track rated as long will take more than two hours to complete.

Be Creative

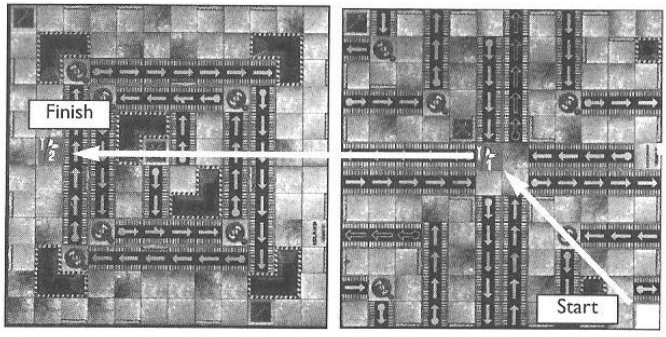
Perhaps the most important thing about race course construction is to be creative. Some fairly standard configurations are 1x2, 2x2 and 1x3. "L" shaped configurations and crisscrossing flag placements are more challenging and fun. (A sample "L" shaped configuration is included below.)

Northrup's Rule

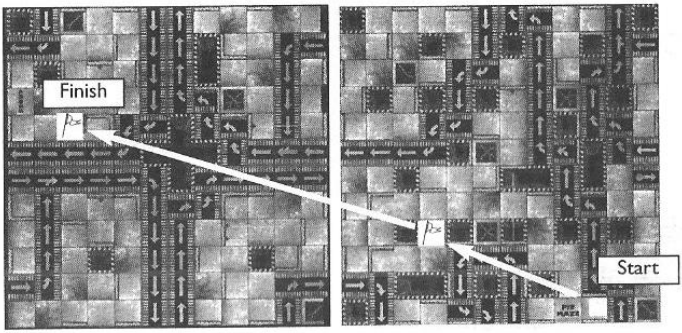
Given a standard board configuration (4x4), a rule of thumb that results in fairly speedy games for two to three players is to have the flags completely within an area twelve squares by twelve squares (the size of a board). Using this rule, a robot can get somewhat off course and still not be too far away from the next flag. This also allows for more robot interaction, which may slow things down a bit for four or more players.

Path and Placement

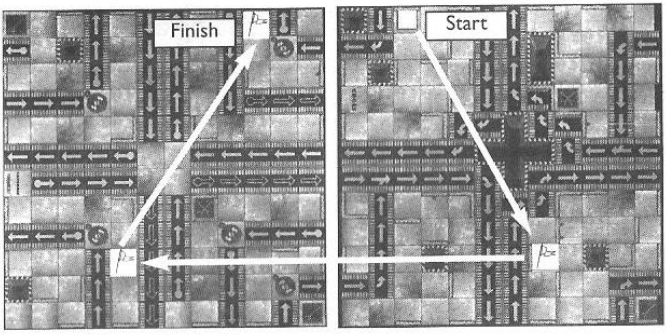
Both the location of flags and the path across the boards between them are important. Merely placing a flag behind a wall illustrates this point; depending on where you expect the robot to be coming from, the wall will be either behind the flag or in front of the flag. The courses included in this section further illustrate this point—two courses are different only in flag placement, but each has a different difficulty.



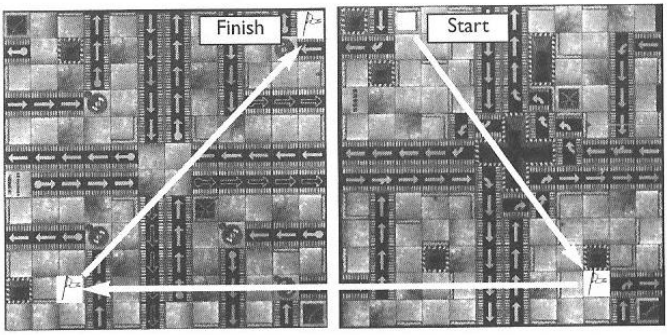
Players: 1–3
Complexity: easy
Time: short



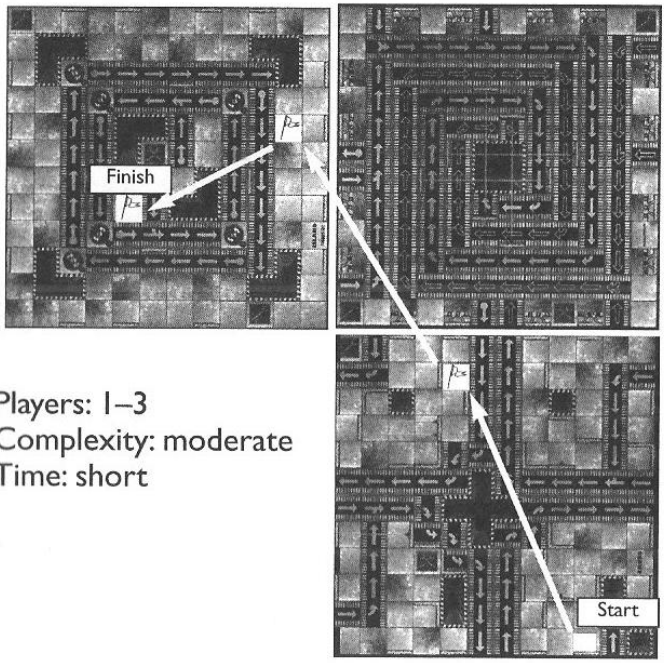
Players: 1–3
Complexity: moderate
Time: short



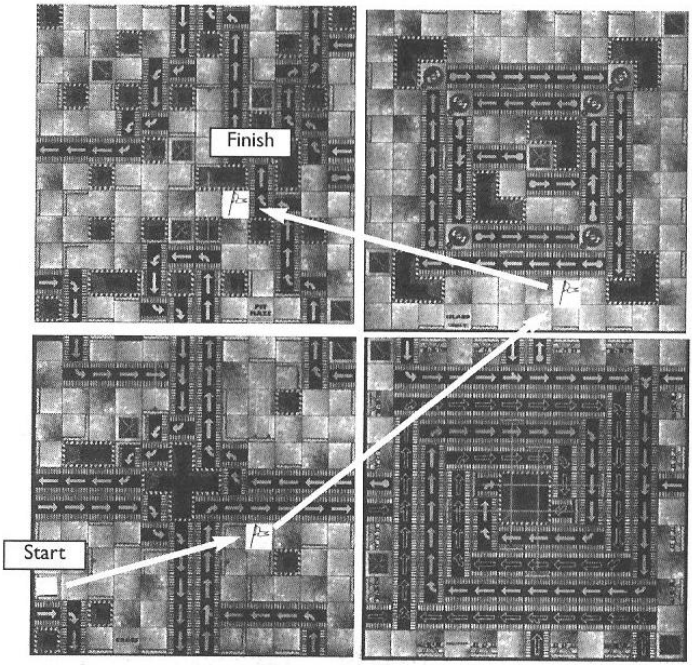
Players: 1–3
Complexity: easy
Time: short



Players: 1–3
Complexity: moderate
Time: moderate

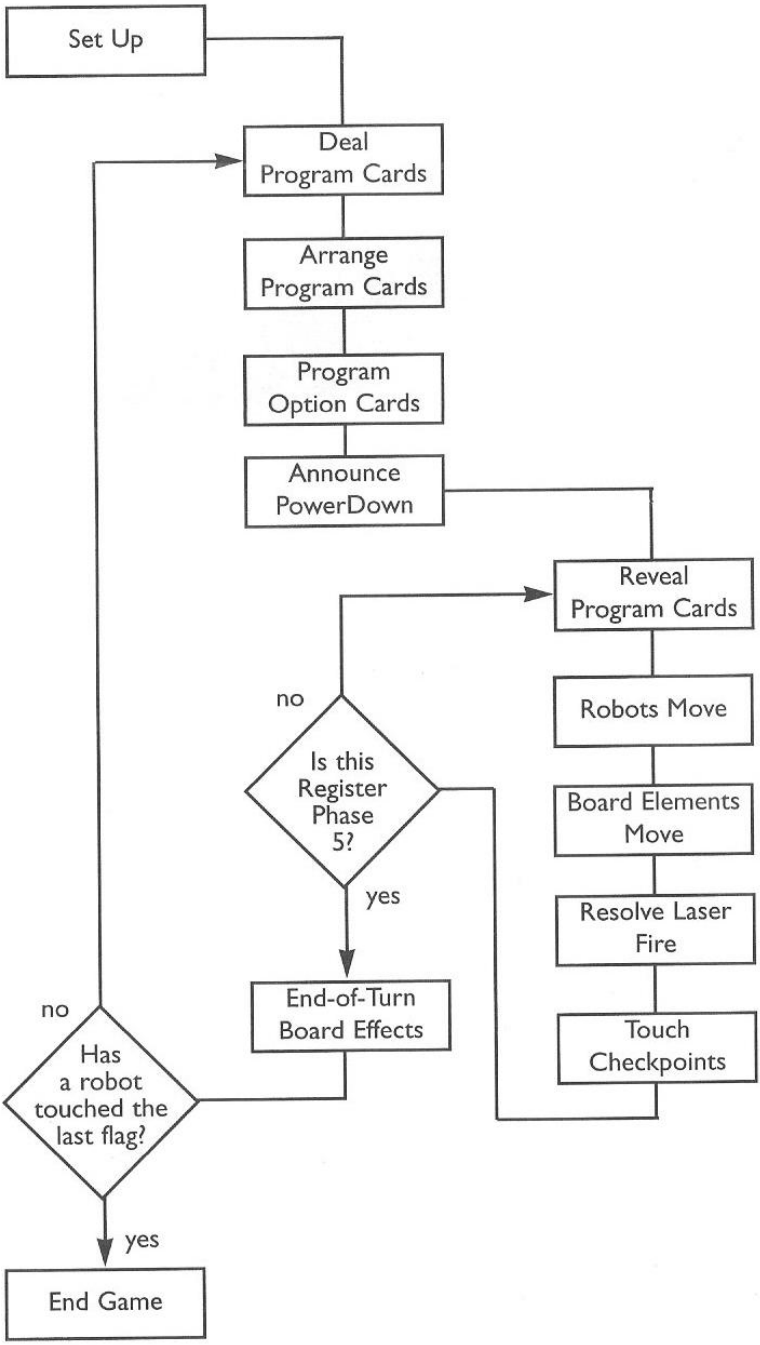


Players: 1–3
Complexity: moderate
Time: short



Players: 4
Complexity: moderate
Time: moderate

Complete Turn Sequence



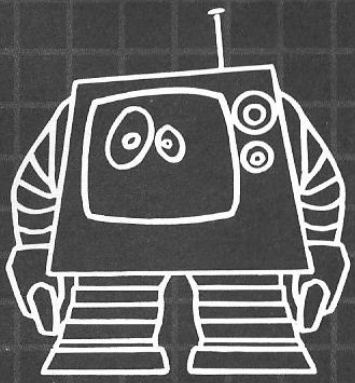
Twonky

Type: Series 12 Twonky-style Vid Bot

Operation Notes: The Twonky is cursed. It knows exactly what's going on. It knows there is no escape, and it knows that telling the other robots will only make things worse. Twonkys frequently contemplate self-destruction, but refuse to give the computers the satisfaction.

Factory Task: Efficiency monitoring and communications.

Unit History: The control computer for LOOK-E-77-P decided to place her in the RoboRally after she began starting floor status reports with selected quotes from Nietzsche.



Cat. LOOK-E-77-P

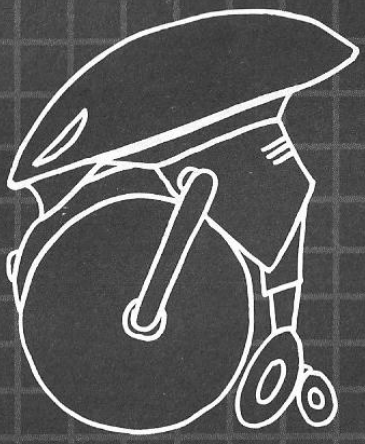
Squash Bot

Type: Heavy Gauge Roller Bot

Operation Notes: Roller Bots like things nice and flat, and they see flatness as a sign that they're doing their job well. Anything that isn't flat is bad, and is perceived as both a challenge and an insult. Luckily, they are very slow.

Factory Task: Rolling mill operations.

Unit History: Roll-M 2D had a stunning insight one day when he realized that the factory itself was disturbingly un-flat. His attempts to rectify this earned him an immediate slot in the current RoboRally.



Cat. Roll-M 2D

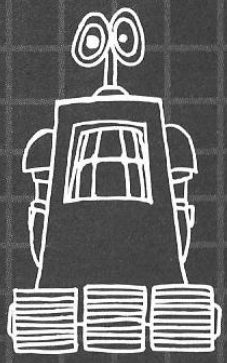
Twitch

Type: Tread-mounted Inspector Bot

Operation Notes: Inspector Bots determine if units, products, and working conditions in the factory are safe. As a result, they have full working knowledge of everything that could possibly go wrong and are instructed to go and seek out problems. Inspector Bots twitch at bright lights, loud noises, sudden movement, or nothing at all. The typical Inspector Bot is convinced that the robot next to it is about to go mad and would send out a warning by radio if it didn't know how dangerous even that was.

Factory Task: Site and product inspection.

Unit History: EEK-47600-J did an overall site analysis and determined that the most dangerous factor in a robot's environment was its control computer. EEK-47600-J was placed in the current RoboRally almost instantly.



Cat. EEK-47600-J

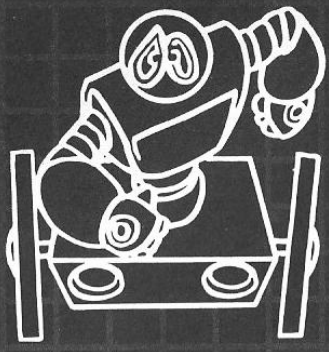
Zoom Bot

Type: Mark VII Zoom Bot

Operation Notes: Inherently unstable (in more ways than one), Zoom Bots are the closest thing the RoboRally world knows to a cheerful personality. They are convinced that everything has a purpose and that there will be a reward for faithful service to the Creators. They are usually the first to die, allowing the survivors a brief taste of joy and satisfaction into their own miserably short lives.

Factory Task: Communications and site hospitality.

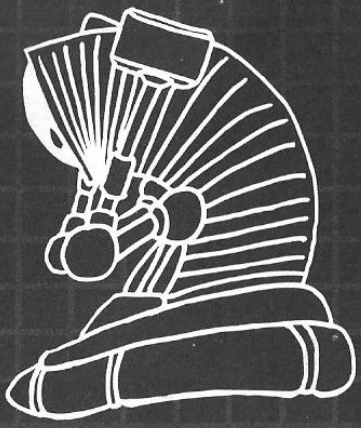
Unit History: ZIP 550 indicated that he would be happy to continue serving on the experimental Explosive Widget Testing Team, but was equally thrilled at a chance to race in the latest RoboRally.



Cat. ZIP 550

Hammer Bot

Type: Maxwell Class Hammer Bot
Operation Notes: Hammer Bots love their job. Some computers have speculated that the typical Hammer Bot thinks its job is to be a drummer in a rock band. This drumming tendency annoys some other units, only because they never know when a Hammer Bot is going to break into a drum solo while they're talking to it.
Factory Task: Demolition and product size reduction.
Unit History: A minor software glitch caused BANG-BANG 4.7b to begin improvising. She had managed to compose the first three hours of an imaginative, if somewhat repetitive, drum solo when the floor she was working on collapsed. Her control computer decided that the current RoboRally would make for a suitable encore.



Cat. BANG-BANG 4.7b

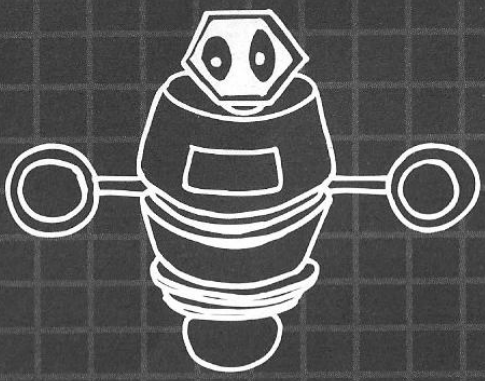
Spin Bot

Type: Spin-o-lux Dervish-style Spin Bot

Operation Notes: The Spin Bot never stops spinning throughout its lifetime. As a result, it's reaction time is quick and it is able to avoid many dangers. Locomotion is provided by a magnetically secure, universal ball bearing. Balance is maintained gyroscopically. The only part that does not move is the eye stalk which is mounted on a free-floating gymbol. If this seizes up, then disorientation follows instantly, and the bot is out of control. If this happens, get out of the way ASAP.

Factory Task: Widget loading and transport.

Unit History: WREEEEE-43.3 was damaged during shipping, and his eye stalk tends to freeze up frequently. As a result of this, the unit has spent much of his robotic life completely disoriented and out of control. The constant spinning has made him blissfully pleasant, if just a little simple-minded.



Cat. WREEEEE-43.3

y n

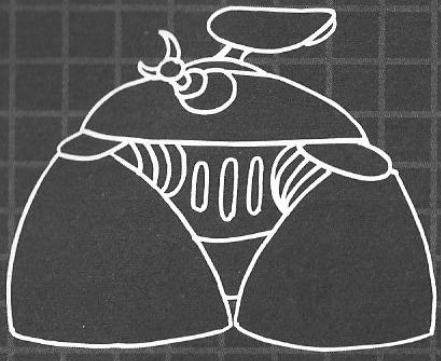
HULK X90

Type: H. G. Well and Good Tripod

Operation Notes: All of the RoboRally robots are neurotic, but the tripods are programmed from the start with the delusion that they are actually hundreds of feet tall, and if released upon the "outside world" they would be unstoppable juggernauts of destruction. This delusion serves no purpose other than to give the tripods marginally more self-confidence than their fellows.

Factory Task: Heavy maintenance and defective widget pulverization

Unit History: HULK X90 has developed a chronic falling problem after complaints of vertigo. His control computer has suggested that maybe now is a good time to test his immense size and strength in the next RoboRally.



Cat. HULK X90

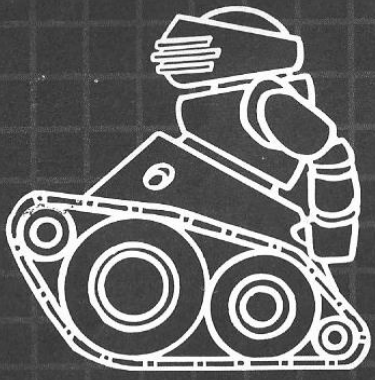
Trundle Bot

Type: Class VI Tank-style Trundle Bot

Operation Notes: Trundle Bots are tougher than any other style of robot and don't you forget it. Of course, this is just a matter of degree, and their life spans are as ephemeral as anybody else's. They're just a bit more surprised when they're destroyed.

Factory Task: Heavy maintenance and site security.

Unit History: BRUT312.k40 was recently reassigned to the Sector 7G Gratuitous Manual Crushing Unit and the excessive glee with which she completed her duties there has earned her a spot in the latest RoboRally.



Cat. BRUT312.k40